File System Quotas [5]

The file system quota enforcement feature allows you to control the amount of file system space in blocks and the number of files used by each account, group, and user on an individual basis. Controls may be applied to some or all of the configured file systems, except for the root file system. Attempts to exceed these limits result in an error similar to the error that occurs if the file system is out of free space. Optional warning levels are also available for informing users when usage gets close to the quota.

Warning: This section contains warnings and information critical to the proper use of a Cray ML-Safe configuration of the UNICOS system.

5.1 Components of the File System Quota Feature

The following components make up the file system quota feature:

- Quota control files (see `quota`(5)).
- Kernel support and enforcement code.
- Quota administration tools (`quadmin`(8) and `qudu`(8)).
- Quota configuration additions to `/etc/fstab` (see `fstab`(5)).
- Quota reporting tool (`quota`(1)).
- Quota warning and limit signal (`SIGINFO`).
- Quota limit error numbers (`EQACT`, `EQGRP`, and `EQUSR`).
- User warning and limit message generation. This feature is provided automatically by the Korn, Bourne, and C login shells. The `quotamon`(1) command is no longer needed.

The quota control files contain all quota information and must be created and maintained correctly. The following sections concentrate on how to set up the quota files and perform the configuration tasks for various operation modes. The quota control files are discussed as though there were one file per controlled file system. Other modes of operation are described in Section 5.15.2, page 263, but you must fully understand the fundamentals of the feature before going on to more elaborate configurations.

Throughout this section, the term *ID* is a generic term meaning account, group, and user IDs. A distinction among the different IDs is needed only when you actually reference specific data structures.

Note: If users on your system use the UNICOS Network File System (UNICOS NFS) to move files, the account numbers will be assigned by the NFS daemon, and the account quotas will not be accurate.

5.2 Enabling the Quota Feature

If you are not using the menu system to set the configuration, find the following line in the `/etc/config/config.mh` file:

```
#define CONFIG_FQUOTAS n
```

Edit this line to set *n* to 1 to enable the quota feature, or to set *n* to 0 to disable it. The kernel must be rebuilt for this change to become effective. For more information on the `config.mh` file, refer to Chapter 4 of the *UNICOS Configuration Administrator's Guide*, publication SG–2303.

If you are using the menu system, you can turn quotas on or off by selecting `on` or `off` in the `Configure System -> Major software configuration -> File quotas` menu of the UNICOS Installation and Configuration menu system. For more information on the UNICOS installation and configuration menu system (ICMS), refer to *UNICOS System Configuration Using ICMS*, publication SG–2412.

These methods apply to all systems that support UNICOS.

5.3 Changing the `NQUOTA` Value

The `NQUOTA` value represents the default number of in-core quota entries. If you are not running file system quotas, the default `NQUOTA` value is `0`. If you are running file system quotas, the default `NQUOTA` value is `1400`. Note that this value must equal `((NINODE/4)` + `NC_SIZE`). If desired, you can change this value to better fit your site's needs.

If you are not using the menu system, you can change the `NQUOTA` value by editing the following line in the `/usr/src/uts/cf.`*SerialNumber*`/config.h` file:

```
#define NQUOTA      1400
```

The kernel must be rebuilt for any changes to become effective. For more information on the `config.h` file, refer to Chapter 3 of the *UNICOS Configuration Administrator's Guide*, publication SG–2303.

If you are using the menu system, you can change the `NQUOTA` value in the `Configure System -> UNICOS Kernel Configuration -> Table Size Parameters -> In core quota entries (NQUOTA)` menu of the UNICOS Installation and Configuration menu system. For more information on the UNICOS installation and configuration menu system (ICMS), refer to *UNICOS System Configuration Using ICMS*, publication SG–2412.

These methods apply to all systems that support UNICOS.

5.4 Quotas and Data Migration

In the default setting, when data migration is turned on and a file is migrated, the space the file occupied is credited to the file owner's ID. When a file is brought back online, the number of blocks is added to the ID's file quota. If bringing a file back would violate an enforced quota limit, that file cannot be brought online.

With the optional aggregate quotas setting, the offline and online date is tracked together, and users are limited in the total amount of space they can use. Because of this, unmigration of a file is always allowed. See Section 5.16, page 263, for more information.

5.5 Configuring Quotas

Information on file system quota configuration is contained in the `/etc/fstab` file (see `fstab`(5)). The instructions for creating quota control files assume that the `/etc/fstab` file has been set up correctly.

Warning: The `/etc/fstab` file is part of a Cray ML-Safe configuration of the UNICOS system. For more information on configuring the `/etc/fstab` file, see the UNICOS Configuration Administrator's Guide, publication SG-2303.

The options field in the `/etc/fstab` file includes a `quota` option, which can be in one of the following three formats. Each of these formats is discussed in one of the following sections.

`quota=`*quota_file_relative_name*

`quota=`*quota_file_full_name*

```
quota=/dev/dsk/filesystem_name
```

5.5.1 Format 1: Relative File Name

Use the first format (`quota=` *quota_file_relative_name*) if you want the quota control file to reside on the file system it controls. The file name is relative to the root directory of the file system and, if you use the default name as recommended (`$QFILE`, as defined in `quadmin`(8)), the option would be written as follows:

```
quota=$QFILE
```

By default, the special name `$QFILE` maps to the `.Quota60` file in the root directory of the file system.

5.5.2 Format 2: Absolute File Name

Use the second format (`quota=` *quota_file_full_name*) if you want the quota control files to reside in a directory other than the root directory of the file system it controls. For example, if the quota files were to reside in the `/etc/admin/quota70` directory, the options field of the `/etc/fstab` file would contain the following line:

```
quota=/etc/admin/quota70/$FILESYS
```

The special name `$FILESYS` is the last component of the file system name on this line in the `/etc/fstab` file. For example, the `/etc/fstab` file contains the following line:

```
/dev/dsk/slash_b    /b    C1FS    quota=/etc/admin/quota70/$FILESYS
```

This line would resolve to the following:

```
quota=/etc/admin/quota70/slash_b
```

A directory, `quota70`, holds all quota control files. The file system name is used to identify each individual quota control file within the directory.

5.5.3 Format 3: Quota Control Groups

Use the third format (`quota=/dev/dsk/`*filesystem_name*) to show that the specified file system is under the control of a quota file defined and used to control other file systems as well. You should use this format when multiple file systems will be controlled as a group. (Multiple `tmp` file systems can be handled this way so that the user's `tmp` quota is independent of which or how many `tmp` file systems might be in use.)

For example, assume that three lines from the `/etc/fstab` file were written as follows:

```
/dev/dsk/tmp_1     /tmp_1     C1FS     quota=$QFILE
/dev/dsk/tmp_2     /tmp_2     C1FS     quota=/dev/dsk/tmp_1
/dev/dsk/tmp_3     /tmp_3     C1FS     quota=/dev/dsk/tmp_1
```

These lines define the quota control file as `.Quota60`, residing in the root directory of `/tmp_1`. The `/tmp_2` and `/tmp_3` file systems are controlled by the same quota control file; therefore, the quota information for usage of any or all of the three file systems is common and reflects the combined usage of all three.

Note: If the right-hand side of a quota option matches one of the other file system names in the `/etc/fstab` file, it is a declaration in the third format, and the specified file system must contain a quota option naming a file. Only one level of indirection is supported.

5.6 Determining Defaults and Special Users

For each file system for which you intend to enforce quotas, you must choose between various quota enforcement options, as follows:

1. Select the class or classes of enforcement you want to use (a combination of user, group, and account quotas).
2. For each of those classes, decide whether to enforce inode quota limits, file quota limits, or both.
3. Pick default values for inode and file limits and their corresponding warning values that will work for most users on the file system.
4. Decide if any special users require limits different from the default values, and decide what those values should be.
5. If you want to use oversubscription, pick one of the evaluation algorithms and an evaluation period. You can perform this task later if you do not want to make a decision now.

All of these choices depend heavily on how the file system is used by users, how many users use the system, and so on; there are no easy rules for making those decisions. However, the `qudu`(8) command allows you to observe current usage on the file system, which helps you choose proper values. After you have picked the values, read Section 5.7, page 244, which shows you how to produce a quota source file for each file system using these values.

The following examples show how to collect both file and inode usage data for the hypothetical file system `/dev/dsk/netos`.

In example 1, the command line prints a list of all user IDs, group IDs, and account IDs that currently have files allocated on the `/dev/dsk/netos` file system. The list is sorted in ascending order, first by ID class and second by number of inodes in use.

Example 1:

```
/etc/qudu /dev/dsk/netos | cut -d' ' -f1-5 | sort +0 -1 +4n
```

In example 2, you want to enforce file quotas only for user IDs. The command line prints a list of user IDs on the `/dev/dsk/netos` file system, sorted into descending order based on the number of disk blocks currently in use.

Example 2:

```
/etc/qudu /dev/dsk/netos | grep uid | cut -d' ' -f1,2,6-8 | sort 4nr
```

5.7 Creating a Quota Control Source File

When a UNICOS kernel has been built with quota enforcement enabled, you must create quota control files for each file system on which you want to enforce quotas. The simplest way to enforce quotas is to place the file in the controlled file system, as described in this section. Other methods are discussed in Section 5.10, page 251. To create a quota file, the information needed in the file must be expressed in `quadmin` directive format (see the `quadmin`(8) man page). You can make a file in the correct format by using `qudu`(8), or you can create your own `quadmin` source file. See Section 5.7.1, page 245, and Section 5.7.2, page 248, for more detailed information on this process.

When you initially create the source file, you must use the `quadmin` command with the `-F` option. The name of the quota control source file is determined from the `/etc/fstab` line belonging to the file system specified on the `filesystem` directive in the `quadmin` input file.

It is not recommended that you create a very elaborate quota scheme the first time. Using appropriate defaults generates quota files almost automatically, and you can always adjust specific quotas later if the need arises.

Note: The file system that is specified and the file system that has the quota control file must be mounted.

5.7.1 The qudu(8) Method of Source Creation

The `qudu`(8) command looks at each inode in a file system and accumulates inode counts and total file size for every account ID, group ID, and user ID found. This information is written in a form suitable for input to `quadmin`(8), which creates or updates the quota control file. When you have run `qudu`, the output file contains usage information for every ID currently using space on the file system you selected. Run `qudu` on one file system and examine the output to get an idea of its appearance and the information it contains. (It takes a few minutes to run this command on a large file system.)

If you do not want usage information for all account, group, and user IDs, select the specific class or classes of ID you want with the `-A` (account), `-G` (group), or `-U` (user) options. If none of these options is present, the default is the same as if `-AGU` had been included. For example, if you want to control only group and user IDs, specify `-GU`.

If you use the resulting output file as input to `quadmin`, a quota control file is created with an entry for every ID in use. The usage information is current; if users are running on the file system, the information becomes obsolete quickly, but every entry has default quotas and warnings assigned to it. If the defaults are satisfactory, you are finished; otherwise, you must change whatever is necessary to get to the state you want.

5.7.1.1 Changing Defaults

If default values are suitable for all IDs using the file system, but the defaults defined in the `sys/quota.h` file are not correct for your site, edit the output from `qudu` and add the correct defaults to the file by using the `quadmin default` directive (see the `quadmin`(8) man page for a description of the directive format). This information should be placed after the `filesystem` and `open` directives and the first `acid`, `gid`, or `uid` directive line in the file. If you use this file as input to `quadmin`, the defaults reflect the values you specified and are applied to every entry in the file.

The following example shows how to use the output file from `qudu` to change default values:

```
#    Usage report by qudu: (SN-1203) on Thu Feb 15 09:36:05 CST 1990
version 6
filesystem dsk/usr_c /usr/c/$QFILE; open dsk/usr_c
```

```
remove all usage

#    The following lines were inserted to change the defaults

     default account file quota 8500 inode quota 700
     default group file quota 10000 inode quota unlimited
     default user file quota 2500 inode quota 300

#    The remainder of the file is exactly as written by qudu

uid 0 inode usage 1375 file usage 25721
gid 0 inode usage 1401 file usage 26165
acid 0 inode usage 7114 file usage 74981
gid 1 inode usage 2 file usage 71
uid 2 inode usage 1 file usage 71
gid 3 inode usage 1 file usage 32
acid 2 inode usage 168 file usage 0
uid 233 inode usage 361 file usage 2500
uid 247 inode usage 1 file usage 0
uid 258 inode usage 12 file usage 2
uid 263 inode usage 334 file usage 5201
uid 264 inode usage 29 file usage 937
uid 269 inode usage 505 file usage 6893
uid 273 inode usage 23 file usage 6548
uid 283 inode usage 1 file usage 0
uid 285 inode usage 29 file usage 0
```

Setting defaults needs to be done only once for each quota control file, because the defaults persist for the life of the file, unless you change them.

Note: You must apply infinite quotas to all account IDs, group IDs, and user IDs used by system daemons, `root`, and other special users. These types of special users should not receive write errors because a quota limit has been reached. Infinite quotas for special users is not part of the software design because the IDs used cannot be predicted. See Section 5.7.1.1, page 245.

5.7.1.2 Setting Specific Quotas

If some IDs in the quota control file need quota values different from the default, follow these steps:

1. Decide on suitable defaults and follow the procedure recommended in the previous section.

2. Using an editor, find the IDs in the file created by qudu that need special attention (they appear in ascending numeric order) and insert appropriate file quota, file warning, inode quota, and inode warning values for the relevant ID class (account ID, group ID, or user ID). If there are only a few IDs that need changing, this process does not take much time. If many IDs require different quotas, this process takes some time, but it is possible to automate the insertion of quota information with a stream editor or a program.

The following example shows how to change quota values for specific IDs:

```
#    Usage report by qudu: (SN-1203) on Thu Feb 15 09:36:05 CST 1990
version 6
filesystem dsk/usr_c /usr/c/$QFILE; open_dsk/usr c
remove all usage

#    The following lines were inserted to change the defaults

     default account file quota 8500 inode quota 700
     default group file quota 10000 inode quota unlimited
     default user file quota 2500 inode quota 300

#    Change the quotas for uid, gid, and acid 0 to unlimited

     acid 0 file quota unlimited inode quota unlimited
     gid 0 file quota unlimited inode quota unlimited
     uid 0 file quota unlimited inode quota unlimited

#    The remainder of the file is exactly as written by qudu

uid 0 inode usage 1375 file usage 25721
gid 0 inode usage 1401 file usage 26165
acid 0 inode usage 7114 file usage 74981
gid 1 inode usage 2 file usage 71
uid 2 inode usage 1 file usage 71

acid 2 inode usage 168 file usage 0
gid 3 inode usage 1 file usage 32
uid 233 inode usage 361 file usage 2500
uid 247 inode usage 1 file usage 0
uid 258 inode usage 12 file usage 2
uid 263 inode usage 334 file usage 5201
uid 264 inode usage 29 file usage 937
uid 269 inode usage 505 file usage 6893
uid 273 inode usage 23 file usage 6548
uid 283 inode usage 1 file usage 0
uid 285 inode usage 29 file usage 0
```

Setting specific quotas needs to be done only once for each quota control file, because the quota settings persist for the life of the file, unless you change them. If you have done much manual work, create an ASCII back-up copy of the quota control file by executing `quota` with the `-b` option.

5.7.2 Manual Source File Creation

You can create a source file for `quadmin`(8) manually, as shown in the example found on the `quadmin`(8) man page. If your installation uses ranges of IDs for specific categories of accounts, groups, or users, this method allows you to create a file fairly easily using the `enable` directive provided by `quadmin`. If each ID has a different quota, you must create directives one at a time. To make this process easier, you can create a skeleton directive file with all of the general information and then edit that file to insert specific quota information. You can always create a source file with a program, if the information needed to specify the quotas for each user is available or can be derived from an existing source.

5.8 Generating the Quota Control File

When the source file has been created, you must generate a quota control file. Use `quadmin` with the `-F` option for this step, because the quota file does not exist and therefore cannot be accessed through the `quotactl`(2) system call. The `-F` option causes `quadmin` to write the file directly.

Follow these steps to generate the file:

1. If the source file has been created initially by `qudu`, and `/etc/fstab` has the current quota configuration, the quota file name on the `filesystem` directive line in the source file should be correct. If it is not correct, choose one of the following methods to rename the quota file:

   - Ensure that `/etc/fstab` is correct and remove the file name from the `filesystem` directive. This forces `quadmin` to use the file system name specified in `/etc/fstab`.

   - Do not have this file system configured in `/etc/fstab`. This forces `quadmin` to use the file system name specified in the `filesystem` directive.

- Change the file system name on the `filesystem` directive line to match the one in `/etc/fstab`. In this case, `quadmin` uses the name from `/etc/fstab`, but, because the name matches the one on the `filesystem` directive, `quadmin` does not warn you about a name mismatch.

Assuming that `/etc/fstab` is set up, the directive line for creating a quota control file in `/usr/c`would read as follows:

```
filesystem /usr/c; open /usr/c
```

Always use an explicit `open` directive so that, if the source files are ever combined, there will be no confusion about what information belongs to each file system.

2. With the proper information in the source file, run the following command:

   ```
   quadmin -F -m source_file_name
   ```

   This command creates the quota control file specified either in `/etc/fstab` or with the optional second parameter of the `filesystem` directive. You must have permission to create files in the directory specified for the quota control file.

3. When you are finished, ensure that the quota file has `root` ownership and owner-only permissions, so other users cannot access or accidentally alter or remove it. `quadmin` creates the file with whatever ownership it has inherited and owner read/write access only.

It is recommended that you create each quota control file separately in order to deal with any mistakes or problems more easily. Also, having a quota source file for each file system makes maintenance easier.

Quotas applied to the root file system or to user ID `0` (`root`) are ignored. The kernel refuses to activate quotas on this file system. Be especially cautious that quota control files do not get removed by accident or inadvertently reloaded from a backup.

5.9 Activating Quota Enforcement

After a quota control file has been created, you can activate quota enforcement on that file system by entering the `quadmin`(8) command with the `-c` option (you must be super user to activate the quota feature; for use on a Cray ML-Safe configuration of the UNICOS system, see the `quadmin`(8) manual page for more information). To activate quota enforcement, you must specify one of the following activation choices as an argument to the `-c` option:

Activation choice	Description
`count`	Turns on quotas for file system; maintains counts.
`default`	Turns on quotas for the relevant file system in the mode it was previously in. The kernel records the activation mode and uses the most recent mode when `default` is used. Unless explicitly changed by use of a `default` directive, a newly created quota file has `count` as the default mode.
`inform`	Turns on quotas for the relevant file system, maintains counts, and issues warning and quota limit signals.
`enforce`	Turns on quotas for the relevant file system, maintains counts, issues warning and quota limit signals, and enforces quota limits.

These choices are described in more detail on the `quotactl`(2) man page.

The following examples use `enforce`, but the other activation choices can be used in the identical manner. The easiest way to activate quota enforcement is to use the quota source file you used to create the quota file. The information in the quota control file has not been changed; `quadmin` searches for the `filesystem` directive in the source file and gets the needed information from it. Run the following command:

```
quadmin -c enforce source_file_name
```

If you do not want to use the directive file, run the following command to activate quota enforcement on the `/usr/c` file system:

```
quadmin -c enforce -s /usr/c
```

If there are no error messages, quota enforcement is running on the file system. You may change the enforcement level at any time.

Quotas may be activated automatically when the file system is mounted if the `-q` option is added to the `mount`(8) command line. The enforcement level is `default`.

5.10 Setting Current Usage Information

The qudu(8) command generates usage information based on the content of the target file system at the time it is run. Because qudu uses the raw device interface, you can run usage extraction on an unmounted device. If the quota control file resides on the controlled file system as recommended, the usage information cannot be entered in the file until the device is mounted. quadmin(8) should be run immediately after the device is mounted in order for the quota control file usage information to be as accurate as possible.

Usage information in the quota control file must be set initially when quota control is installed on the file system and whenever fsck(8) changes information related to usage. (You will not know when fsck changes information, so run qudu routinely after fsck.) Inaccurate usage information in the quota control file does not affect system processes, but quota limit and warning thresholds are dependent upon the accuracy of the quota control file. To ensure that quota control is consistent and accurate, make it a policy to update usage information before users can alter the file system.

5.11 Usage Accumulation Rules

Assuming that both inode and file space are controlled, the following sections describe how the kernel accounts for space usage.

5.11.1 Inode Usage

Only inodes associated with regular files (IFREG), migrated files (IFOFL, IFOFD), symbolic links, and directories (IFDIR) are counted. The account, group, and user IDs found in the inode are charged one unit for each inode belonging to them. Inodes are not counted as file space.

5.11.2 File Usage

Any indirect blocks (blocks containing disk allocation information not accessible to the user) are counted as file space, as are access control limit (ACL) blocks, if the UNICOS multilevel security (MLS) feature is enabled.

Files and directories whose data resides entirely in the inode (possible only on CRAY T90, CRAY C90, and CRAY Y-MP systems) have an allocated block count of 0. Only actual blocks allocated are counted, rather than the space logically used. Therefore, the length reported by the ls(1) or du(1) commands can be very different from the amount of space really used (either more or less), especially if sparse allocation occurs or files have data residing in the inode. The amount charged is always in units of allocation rather than logical length, which varies depending on the implementation and device configuration from one sector to many tracks per unit.

When files are migrated, the space occupied is credited to the IDs in the inode of the file. If the file is later brought back online, the space needed for the file is applied against the quotas. If bringing a file online violates an enforced quota limit, that file cannot be brought online.

5.12 Administering the Quota Enforcement Feature

The following sections contain information about basic administration of the quota feature, including starting up the system, activating quota enforcement control, adding users, deleting users, creating or extending files, and viewing network quota information.

5.12.1 System Startup

If you run `fsck`(8), also run `qudu`(8) to generate correct usage information. Immediately after the file system is mounted, run `quadmin`(8) to correct the quota control file.

The following example shows how the file system quota feature can be activated at system startup. The first box contains a script from the `/etc/rc.mid` file, which is called by the system start-up script file `etc/rc`. (The script can be placed in any file called by the system start-up script.) In this example, the script uses the shell script `/admin/etc/quotas/qurun` (shown in the second box) to create and update each file system's quota control file, start the file system quota enforcement feature, and set the enforcement level to `enforce`.

```
#
#   initialize quotas for the day
#
               if [ -f /admin/etc/quotas/qurun ]
               then
                  echo "Rebuilding disk quotas... (takes 7 minutes)..."
                  /admin/etc/quotas/qurun -i
                  echo "Disk Quotas rebuilt"
               fi
        fi
        echo ''
#
```

```
#   turn quotas on and run the exceptions files
#
/admin/etc/quotas/qurun -s enforce
/admin/etc/quotas/qurun -d
```

In this example, the `/admin/etc/quotas/qurun` shell script is used to automate routine administrative duties for each file system specified on the shell script's command line. If no file systems are specified, the list `$FILESYSTEMS` is used. The options that can be specified on the command line are as follows:

Option	Description
`-b`	Backup. For each file system, this option creates a back-up file in directory `$BACKDIR` containing all the `quadmin` directives necessary to reconstruct the file system's current quota file, excluding usages. The quotas feature can be running, and users can be in the file system during a backup.
`-i`	Install. For each file system, this option creates a new quota control file by using the defaults in the back-up file (if it exists) in directory `$BACKDIR`. The `qudu` command is then used to update the quota file with the current usages. This option is useful when setting up quotas for the first time, or when the quota file has been destroyed or is out-of-date. The quota feature cannot be running during an install, and users cannot be in the file system.
`-d`	Defaults. For each file system, this option updates the quota control file with any defaults from the back-up file in directory `$BACKDIR`. This option is useful when you have made changes to the back-up file and you want to activate those changes in the file system. The quotas feature can be running, and users can be in the file system when the quota control file is updated.
`-r`	Report. For each file system, this option issues a report showing the number of disk blocks and inodes owned by each user, group, and account, sorted by increasing usage amount. This option is useful when picking defaults or when looking for users who have exceeded their limits. The quotas feature can be running and users can be in the file system while a report is issued. The `-r` option requires that the `awk` script `qusort` be in your binary search path. (The `qusort` script is intended to provide a mechanism for formatting the `qurun` report; it is not provided in the release or in this example.)

`-s`	Start. For each file system, this option starts the quotas feature or, if the feature is already running, it changes the enforcement level. The available levels are `count`, `inform`, and `enforce`. If you do not specify a level with `-s`, the `enforce` level is used. To preserve accuracy, you must make sure no users are in the file system when turning on the quotas feature. Users can be in the file system when changing enforcement levels.
`-u`	Update. For each file system, this option uses the `qudu` command to update the quota file with the current usages. This option is primarily used while the quota feature is running to update the quota file when it is believed that the quota file is out-of-date. The update option improves the accuracy of the usages, but it does not guarantee absolute accuracy.

The following box contains the shell script `/admin/etc/quotas/qurun`.

```
FILESYSTEMS="/sn1001/soft/os /sn1001/cnn /sn1001/mktg"
                                   # default list of filesystems to act on
BACK51=/admin/etc/quotas/backup    # directory for 5.1 backup files
BACK60=/admin/etc/quotas/backup60  # directory for 6.0 backup files
QUOTDIR=                           # directory for quota files

BIN=/etc/       # directory where the quota commands reside
#
#       Determine the operating system level.  Use it to pick the correct
#       backup directory.
#
${BIN}quadmin -Q >/dev/null 2>&1
if [ $? -eq 0 ]; then           # if at 6.0 or greater
        QFILE=.Quota60
        BACKDIR=$BACK60
else
        QFILE=.Quota51
        BACKDIR=$BACK51
fi

flags=0
set -- ‘getopt ’bdhirsu’ $*‘if [ $? -eq 0 ]; then
    while [ "$1" != "--" ]
    do
        case $1 in
        -b)
            command=Backup;;
        -d)
            command=Defaults;;
        -i)
            command=Install;;
        -r)
            command=Report;;
        -s)
            command=Start;;
        -u)
            command=Update;;
        esac
        shift
        flags=‘expr $flags + 1‘    done
    shift
fi

if [ $flags -ne 1 ]; then
    cat >&2 <<EOF
Usage: ‘basename $0‘ [ -b | -i | -d | -r | -s [count|inform|enforce] | -u] [
filesys ...]
‘basename $0‘ is used to administer the File Quotas feature.
EOF
    exit 1
fi

if [ "$command" = "Start" ]; then   # if Start, check for enforcement level
    level=enforce
    case $1 in
    count|inform|enforce)
        level=$1
        shift
    esac
fi

if [ $# -ne 0 ]; then       # if no filesystems specified, use default list
    FILESYSTEMS=$*
fi

for filesys in $FILESYSTEMS
do
    echo $filesys
    DEVDSK=‘/etc/mount | awk ’$3 == "’$filesys’" {print $1}’‘    if [ "$DEVDSK" = "" ]; then
        echo "$filesys is not mounted!  $command not done."
        continue
    fi
#
#   Determine if quotas is running on the filesystem.  Sometimes it matters.
#
    QUOFF=F                                     # assume quotas not running
    ${BIN}quota -s $filesys 1>/dev/null 2>&1
    if [ $? -eq 0 ];then
        QUOFF=                                  # quotas is running
    fi
#
#   Compute the backup file and quota file names for this filesystem.
#
    bakfile=$BACKDIR/‘basename $filesys‘    if [ "$QUOTDIR" ]; then
        quofile=$QUOTDIR/‘basename $filesys‘    else
        quofile=$filesys/$QFILE
    fi

#   Perform the requested subfunction.
#
    case $command in
    Backup)
        rm -f $bakfile
        if [ ! -r $quofile ]; then
            echo "Quota file $quofile not found.  Will create it now."
            $0 -u $filesys
        fi
        ( ${BIN}quota -bEs $filesys -q $quofile > $bakfile ) &
        ;;
    Defaults)
        if [ -r $bakfile ]; then
            ( ${BIN}quadmin -m$QUOFF $bakfile ) &
        else
            echo "File $bakfile is unreadable; Defaults are unchanged.">&2
        fi
        ;;
    Install)
        if [ "$QUOFF" ]; then
            rm -f $quofile
            $0 -d $filesys
            ( ${BIN}qudu -q $quofile $DEVDSK | ${BIN}quadmin -mF ) &
        else
            echo "$filesys has quotas enabled!  Install not done."
        fi
        ;;
    Report)
        echo "                        filesystem $filesys"
        ${BIN}qudu $DEVDSK | ${BIN}qusort #Run output through qusort to format
        echo ""
        ;;
    Start)
        if [ ! -r $bakfile ]; then
            echo "Backup file $bakfile not found.  Will create it now."
            $0 -b $filesys
        fi
        ${BIN}quadmin -c $level $bakfile
        ;;
    Update)
        ( ${BIN}qudu -q $quofile $DEVDSK | ${BIN}quadmin -m$QUOFF ) &
        ;;
    esac
done
wait
```

If you decide to use the example script, you must first set the first three variables to match your configuration. The variables are as follows:

Variable	Description
`FILESYSTEMS`	Set to the list of file systems on which you intend to run quotas. The file system names should be separated by blanks.
`BACKDIR`	Set to the path name of a directory where back-up copies of each quota file will be kept. The back-up files contain all the default quota settings and all settings for users whose limits are different than the defaults.
`QUOTDIR`	Usually, leave this blank, which causes the quota file for each file system to be placed in the root directory of that file system. If you are short of disk space in the quota file systems, set `QUOTDIR` to the path name of a directory on a file system where there is more room and where all the quota files are kept. If you install quotas and then later decide to change `QUOTDIR`, make sure that

you place the new quota file path names in the `filesys` directive of each of your back-up files.

5.12.1.1 Back-up File Example

The following box contains an example of a back-up file. The back-up file is located in directory `$BACKDIR` and is created by `/admin/etc/quotas/qurun` option `-b`. The back-up file is used by options `-i` (install) and `-d` (defaults).

```
# Created by quota on Thu Aug 31 11:54:48 1989
filesystem /sn1001/soft/os /sn1001/soft/os/.Quota60; open /sn1001/soft/os
default account flags off
default account file quota 40000 file warning 0.900000
default account inode quota 200 inode warning 0.900000
default group flags off
default group file quota 40000 file warning 0.900000
default group inode quota 200 inode warning 0.900000
default user flags fi
default user file quota 4000 file warning 0.900000
default user inode quota 1000 inode warning 0.900000
user root file quota infinite inode quota infinite
```

5.12.2 Quota Enforcement Control

When a file system is mounted by using `mount`(8), quota control is activated in its default mode. If a `default level` directive is not included when the quota control file is created, the default mode is `count`. The enforcement mode can be changed by using the `quadmin -c` option, and the specified mode then becomes the default mode. After quota control is activated, you can change its mode, but you cannot deactivate it; only unmounting the file system by using `umount` (see `mount`(8)) deactivates quota control.

5.12.3 Adding Users

If a new user of the file system becomes active, the kernel automatically creates quota control entries with default quota and warning values for any IDs not already defined in the quota control file. If the default values are improper, you must run `quadmin` when you create the new IDs to set up the correct quota information on all file systems available to those IDs. When you run `qudu` to set up initial usage information, `quadmin` adds records to the quota control file with default quota and warning values for all IDs found on the file system, whether or not they were defined in the existing quota control file.

5.12.4 Deleting Users

You can use `quadmin` to delete entries from the existing quota control file. When an entry is deleted, its usage value is set to `0` and the limit and warning values are set to their defaults; the entry is not removed.

To remove an entry from the quota control file, use the following steps:

1. Create the ASCII version of the quota control file by using the `quota` command.

2. Using an editor, delete all records corresponding to the deleted entry.

3. Create a new quota control file by using the `quadmin` command.

   Note: If any inodes with the deleted or removed ID exist in the file system, those entries are restored (with default limit and warning values) if the output of `qudu` is processed by `quadmin` without any editing.

5.12.5 Creating or Extending Files

One of the following rules is applied when files are being created or extended. Both inode and file space quotas are dealt with at the same time (assuming both are being enforced). The privilege of the process making the request is not an issue, so processes owned by `root` are constrained by the same rules as others.

- If the file is owned by `root`, quotas are not enforced, and the file can be extended without limit.

- If the file is not owned by `root`, the owner's quotas are enforced.

When ownership of a file changes, one of the previous rules is followed, depending on the target's ID.

5.12.6 Viewing Quota Control

The `quadmin`(8) command executed with the `-v` option displays generic quota information and the enforcement level of each file system with active quota enforcement. The following example shows a typical display created by `quadmin -v`:

```
Quota Generic Information on Fri Feb  9 10:17:51 CST 1990
Quota control configured for: Accounts, Groups, Users
Quota entries configured:       375
        Maximum used:             78, Current:       36
```

```
Quota entry accesses: get     487532,    put     487294
Quota file accesses: read      54219, update       1004
Quota hash chain:    read         34
Inode cache flushes:               0,  sleep          0

/c.......       INFORM     /d.......COUNT    /fsmtest..ENFORCE
```

This display created by `quadmin` shows that the kernel is configured to support account, group, and user quotas and has a total of 375 quota entries configured. The `Maximum used` and `Current` fields show the greatest number of quota entries used at one time and the current number in use. The maximum value is intended to help you decide how large the quota table in the kernel must be. If the maximum is far below the number of configured entries and stays that way over a period of time when you know typical workloads are being run, you may reduce the number configured, thereby reducing the size of the kernel.

The number of hash chain reads indicates how many quota records were found other than at the beginning of a hash chain. This number is not important in most situations, because it is mostly an indication of the hashing mechanism's efficiency relative to the particular mix of IDs being used.

Usually, tuning the configuration too close to the minimum is not advisable. Individual quota table entries are not very large; reducing the total number by a small percentage does not result in any significant benefit. If you see consistent behavior such as that indicated in the example display (where 375 entries are configured and the maximum usage is only 78), and it would be advantageous for your kernel size to be reduced slightly, you can safely lower the number of configured quota entries to 150.

The remainder of the statistics portion of the display shows information that is useful only in extreme situations. The `get` and `put` values show how many times quota table entries were read and changed. The `read` and `update` counts show the actual number of disk accesses needed to retrieve new quota entries and to keep the file current.

The ratio between `get` and `read` or between `update` and `put` indicates how beneficial cache is in reducing actual I/O activity caused by quota enforcement. If any I/O errors occur on the quota control files, an error count also appears in the display.

The last statistics line shows the number of times the quota system was required to flush inodes and/or sleep in order to acquire a quota table entry for a newly opened file. These numbers should normally be small; significant increases indicate an insufficient number of quota entries. If it persists, this condition can reduce system performance.

The last line on the display lists the file systems and their level of quota enforcement. Only file systems with some level of quota enforcement are shown, and the format is adjusted to fit the maximum number of names on a line consistent with an attractive and readable format.

5.13 Additions to Login Profile

If you want automatic reporting of quota warning and limit conditions when the user logs in or a batch job starts, place the `quota`(1) command in the login profile (`.login` or `/etc/cshrc` (see `cshrc`(5)) for `csh`(1) and `.profile` or `/etc/profile` (see `profile`(5)) for `ksh`(1)), as follows:

```
quota -r wl
```

This command line reports the names of any file systems where the user ID, default account ID, or default group ID have usage values above the warning level. If you want to check all users' authorized account IDs and group IDs, add the following command to the login profile:

```
quota -A -G -r wl
```

If you only want to report on the user's home file system, add the `-s` option to `quota`, specifying the path of the home file system.

Because `quota` provides special exit codes for specific conditions, you can write scripts to analyze the information and produce more informative messages, provide information to local log files, inform the administrator, or perform special actions if batch jobs are involved. Refer to the `quota`(1) man page for more information.

5.14 User-level Behavior

User warning and limit messages are automatically written to the `stderr` file by the Korn, Bourne, and C login shells. However, if you are not using any of these shells, and if you want to enable user warning and limit messages, you must start one copy of `quotamon`(1) for each session (interactive and batch). Without it, users are not notified of quota warning conditions unless they use `quota`(1) periodically. Before using `quotamon`, decide whether the cost of running this background process is acceptable in your environment; `quotamon` is small and uses no CPU time except when writing messages.

If you are running a Cray ML-Safe configuration of the UNICOS system, prior to changing your active security label you must kill `quotamon`, because it is a background process. After changing your label, you can restart `quotamon`.

Starting `quotamon` is best done by adding code to `/etc/profile` and `/etc/cshrc`, as in the following:

```
ps | grep quotamon > /dev/null
if ($status == 1) then
   /usr/bin/quotamon -s 60
endif
```

The `-s` option specifies the amount of time in seconds to delay repetitions of the same kind of quota message; the default is 45 seconds. You may specify as large a value as you wish (up to the signed integer maximum).

The `quotamon` process can be killed by the user. This capability allows users to remove the process if they object to its presence.

If a program traps a `SIGINFO` signal, you can use the `getinfo`(2) system call to determine what event occurred. If a `SIGINFO` occurs, the signal is sent to all processes attached to the same job table entry that have not cleared a previous signal occurrence by executing a `getinfo` system call. The signal is ignored by default.

5.15 Nonstandard Configuration Options

This section discusses alternative ways to configure quota control files; the recommended configuration provides for one quota control file per file system, with each such file residing on the file system it controls (all configuration options are provided through the `/etc/fstab` file).

The kernel can activate quota control where the quota control file is on another mounted file system and also allows more than one file system to be controlled by the same quota control file. However, you must understand the consequences of nonstandard configurations before using them.

5.15.1 Nonresident Quota Control Files

The major operational complications resulting from configuring a quota control file that does not reside on the file system it controls are as follows:

- The file system on which the quota control file resides must be mounted before quotas can be activated on the controlled file system. Conversely, you must unmount the controlled file system before unmounting the file system on which the quota control file resides.

- You should place the configuration information in `/etc/fstab` to ensure that quota usage is consistent. You can use the `-Q` option with the `mount`(8) command to perform this function, but this is not recommended.
- Performance may be affected by too much quota control file traffic on a single file system.
- Quota control would be lost on the controlled file system (but the system would continue to perform I/O correctly) if the control file became inaccessible because of some malfunction on its file system.

5.15.2 File System Groups under One Quota Control File

File system groups under control of one quota control file (also known as *quota groups*, *domain quotas*, or *quota domains*) are similar in operational complexity to the configuration described in the previous section. The key to successful use of this feature is placing the configuration information in the `/etc/fstab` file so that all administrative commands have a consistent view of the organization. Usage information generated by `qudu`(8) consists of the sum of usage for all IDs present on all the file systems that are members of the quota group when any one file system is specified on the command line.

5.16 Aggregate Quotas

The aggregate quotas option enables sites to charge for offline files in much the same way as they charge for online files. In the default quotas setting, an attempt to migrate a file could fail because bringing the file online might violate a quota limit. With the aggregate quotas setting, file unmigration is always allowed by the quota mechanism. The UNICOS operating system tracks offline and online data together, and users are limited in the total amount of space they can use.

Using aggregate quotas has the following ramifications:

- Administrators can limit (approximately) the amount of space used in the offline archive.
- Administrators have no quota control over what users have online; they can control only the total space used.

- Offline file data blocks are always charged in units of 4096-byte blocks; therefore, the quota charge for an offline file is always less than or equal to the charge for the same file online, depending on how the file was created or how the file system is configured.
- Users can accidentally unmigrate more data than the file system can hold.
- Users are always charged for data blocks and MLS blocks, whether or not the file is migrated.

Aggregate quotas affect the following commands:

`qudu`(8)	The `-a` option must be used to specify to use the aggregate quota counting method.
`quadmin`(8)	The default style directive must be changed from `online` to `aggregate`. This directive specifies whether quota counts are maintained only for files that are physically on disk (online), or whether files migrated offline by the Data Migration Facility (DMF) are included in the count (aggregate).
`quota`(1)	The quotas report column *File blocks* is changed to *Aggregate blocks*. This column shows the quota block counts and block usage for aggregate quotas.

5.17 Using the Oversubscription Option (Soft Quota)

The normal operation of the file quota mechanism is to issue warning messages to the user when the warning level is exceeded and to stop any new allocation if the quota is reached. In addition to this mode of operation, an oversubscription mechanism is also available on a file system basis. This mechanism lets users exceed quotas by a controlled amount for a limited period of time.

5.17.1 Behavior of the Oversubscription Mechanism

Each quota control file has header fields that select the oversubscription algorithm to be used and specify parameters to the algorithm. This means that all account IDs, group IDs, and user IDs controlled from the file are under a single discipline. Oversubscription is available for file space only; inode use is enforced with the standard mechanism. When you select this mode of operation by setting the algorithm selector in a header field, quota control changes its behavior as follows:

- The quota remains the hard upper boundary on allocation.

- A second field in the quota record, `f_runquota`, becomes the enforced quota value. When this field is nonzero, the kernel enforces at this value rather than at the `f_quota` allocation limit.
- The warning level becomes the oversubscription threshold.
- The `quota`(1) command changes its display to show information related to the soft quota, such as the time when new allocation is prohibited or permitted.
- The `quadmin`(8) command supports additional fields needed by the evaluation algorithms in the header and record structures.

5.17.2 Supported Algorithms

Two oversubscription algorithms, an exponential and a linear function, are supplied, and two additional algorithm selection values are reserved for site use.

The default algorithm (named `none`) realizes the original behavior of file quotas. Sites that have access to source can add local algorithms (named `site1` and `site2`) to the kernel and their inverses to the `quota`(1) command. The `quota`(1) command uses the inverse to provide predictive information to the user.

5.17.2.1 Exponential Algorithm

The exponential algorithm, as follows, is based on the COS RDM oversubscription model.

$$A = U_s + (A_s - U_s)e^{\frac{-(t-s)}{P}}$$

where:

A = Average now
A_s = Average at time s
P = Characteristic period (in seconds)
s = Time at A sub s
t = Time now
U_s = Usage at time s

In Table 32, the second column is the name used by the `quadmin`(8) command to set the field. Note that the implementation is such that if usage changes from one calculation period to the next, *U s* is set to current usage, *s* is set to the current time, and *A s* is set to *A*. This prevents usage changes from affecting the average until at least some history of that usage level can be accumulated.

Table 32. Field usage of the exponential algorithm

Location	Field name	Variable	Description
Header	`algorithm`		Exponential (actual value)
Header	`ef1`	P	Characteristic period in seconds
Header	`ef2`		Not used
Record	`ef1`	A_s	Average at time *s*
Record	`ef2`	U_s	Usage at time *s*
Record	`ef3`		Not used
Record	`ef4`		Not used
Record	`ef5`	s	Prior evaluation time

5.17.2.2 Linear Algorithm

The linear algorithm is a linear form of oversubscription that eliminates the more time-consuming calculation of the exponential function.

$$A = U_s \frac{(t-s)}{P} + A_s \frac{P-(t-s)}{P}$$

where:

A = Average now
A_s = Average at time s
P = Characteristic period (in seconds)
s = Time at A sub s
t = Time now

U_s `= Usage at time` s

In Table 33, the second column is the name used by the `quadmin`(8) command to set the field. Note that the implementation is such that if usage changes from one calculation period to the next, U_s is set to current usage, s is set to the current time, and A_s is set to A. This prevents usage changes from affecting the average until at least some history of that usage level can be accumulated.

Table 33. Field usage of the linear algorithm

Location	Field name	Variable	Description
Header	`algorithm`		Linear (actual value)
Header	`ef1`	P	Characteristic period in seconds
Header	`ef2`		Not used
Record	`ef1`	A_s	Average at time s
Record	`ef2`	U_s	Usage at time s
Record	`ef3`		Not used
Record	`ef4`		Not used
Record	`ef5`	s	Prior evaluation time

5.17.2.3 Algorithm Comparison

Both the exponential algorithm and the linear algorithm have similar long-term behavior, but the linear form generally reacts more quickly to drastic changes in usage. The inverse functions mentioned previously are used to predict when inflation or decay will cross the warning value (soft limit) and so answer the following questions:

- When will I be able to allocate more space if the average is above the soft limit but actual usage is below?
- When will I be prevented from allocating more space if the average is below the soft limit but actual usage is above?

5.18 Changing ID Class Control

If control over an ID class (account, group, or user) is added or deleted while quota enforcement is active on a file system, files whose inodes are in the cache before the change is made are not affected and retain their prior class enforcement state. This happens, for example, if a file system has been set up with `default account flags off` and then, while the system is mounted, a `default account flags fi quadmin` directive is issued. This behavior should be taken into account if you change the class enforcement policy on an active file system.

5.19 Quota Enforcement Across a Network

When dealing with the control of file space consumed by processes running as servers to another machine, quotas can be enforced only on the machine that physically owns the resource. This situation occurs because quota enforcement on a file system must be done in one and only one place, and the machine that owns the resource presumably has administrators most interested in its control. Also, in heterogeneous networks, there is no guarantee that any node other than the one owning a resource really knows what it is, how it is allocated, and so forth.

A question you may have about this situation is how much information about the resource consumer is available to the enforcing node. To enforce quotas, the node must know the relevant IDs of the consumer, but in situations where IDs are mapped onto the node there is a potential for ambiguity. Also, UNICOS supports account IDs, but most network protocols, including NFS and OSI, do not have the capability of passing an account ID to the server.